UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/601,439	06/23/2003	Dz-Ching Ju	20002/16406	9124

34431 7590 05/31/2007

HANLEY, FLIGHT & ZIMMERMAN, LLC
150 S. WACKER DRIVE
SUITE 2100
CHICAGO, IL 60606

EXAMINER	
YIGDALL, MICHAEL J	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
05/31/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/601,439	JU, DZ-CHING	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 March 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *10/5/06*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is responsive to Applicant's submission filed on March 5, 2007. Claims 1-24 are pending.

Response to Arguments

2. Applicant's arguments have been fully considered but they are not persuasive.

Applicant contends that "Ju does not describe or suggest that the eager mode instruction is inserted in response to determining that the software instruction is an excepting instruction," and that "Ju inserts the instruction prior to determining that the instruction has generated an exception because Ju is directed to determining if the inserted instruction has generated an exception during execution, in direct contrast to determining that the replaced instruction is capable of generating an exception" (remarks, pages 10-11, emphasis in original).

However, Ju's teachings show determining that an instruction is an "excepting instruction" and inserting a control speculative version of that instruction, such as recited in the claims. First, the examiner notes that while a reasonable interpretation of an "excepting instruction" is an instruction that generated an exception, or an instruction that is presently generating an exception, Applicant's arguments imply that the meaning of an "excepting instruction" is an instruction that is *capable* of generating an exception. This is consistent with Applicant's specification, which states, "An excepting instruction is an instruction that may cause an exception to occur" (specification, paragraph [0025]).

Indeed, where Ju teaches determining whether the load instruction generated an exception (see, for example, column 23, lines 27-36), Ju confirms that the load instruction is in fact *capable*

of generating exceptions. In other words, Ju confirms that the load instruction is an "excepting instruction." Ju further teaches inserting an eager mode version of the load instruction (see, for example, column 23, lines 9-14), which is how Ju "handle[s] control speculative relocation of the load instruction" (column 22, lines 65-67). Thus, Ju teaches inserting a control speculative version of the load instruction.

Furthermore, the eager mode instruction is inserted "in response to" the determination that the load instruction is an excepting instruction at least in the sense that in Ju, it is *already* determined that the load instruction is capable of generating exceptions. Ju's method of control speculation specifically considers the fact that the load instruction is capable of generating exceptions (see, for example, column 22, lines 3-13). Thus, Ju teaches inserting a control speculative version of the load instruction in response to determining that the load instruction is an excepting instruction. The examiner respectfully submits that the language of the claims does not patentably distinguish over the teachings of the reference.

Applicant contends that Ju does not describe or suggest determining that an instruction is not an excepting instruction (remarks, pages 11-12). The basis of Applicant's argument is that "an add instruction is inherently an excepting instruction" (remarks, page 11).

However, the examiner respectfully submits that Applicant's remarks are not sufficient evidence to support a conclusion of inherency. As Applicant acknowledges, Ju does not indicate that the add instruction ever generates an exception (remarks, page 11). Instead, Applicant postulates that "an add instruction is capable of triggering an arithmetic overflow exception," and referring to paragraph [0025] of Applicant's specification, states, "An instruction that is capable of triggering an arithmetic overflow exception is an excepting instruction" (remarks, page 11).

First, the examiner notes that Applicant's specification states merely, "In an exemplary embodiment, arithmetic overflow is an exception that could be generated by a multiplication instruction" (specification, paragraph [0025]). Here, the specification describes a multiplication instruction, not an add instruction. Furthermore, Applicant's specification is not a part of the reference. What is possible or probable in the present application is not necessarily the case in the reference. The teachings of the reference itself, without the benefit of Applicant's specification, do not support a conclusion that the add instruction *must* be capable of generating an exception. As noted above, Ju expressly teaches that the load instruction is capable of generating exceptions, and in direct contrast, makes no suggestion that the add instruction ever generates an exception.

The examiner notes that Applicant's specification does not provide any apparent examples of an instruction that is not an excepting instruction.

Applicant's other arguments (remarks, pages 12-16) are analogous to those addressed above and are therefore considered unpersuasive for the same reason(s).

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1-4 and 10-19 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 6,260,190 to Ju (art of record, "Ju").

With respect to claim 1 (currently amended), Ju discloses a method of preserving exceptions in code reordering (see, for example, column 22, lines 3-11, and column 23, line 67 to column 24, line 4), the method comprising:

receiving a plurality of software instructions including a software instruction at a first location within the plurality of software instructions (see, for example, column 22, lines 43-51, which shows receiving a plurality of instructions including a load instruction at a first location);

determining that the software instruction is an excepting instruction (see, for example, column 23, lines 27-36, which shows determining that the load instruction is an excepting instruction);

inserting a control speculative version of the software instruction at a second location within the plurality of software instructions in response to determining that the software instruction is an excepting instruction (see, for example, column 23, lines 9-14, which shows inserting an eager-mode or control speculative version of the load instruction at a second location);

replacing the software instruction at the first location with a check instruction at the first location (see, for example, column 23, lines 14-16, which shows replacing the load instruction at the first location with a check instruction); and

generating a recovery block which branches from the check instruction, the recovery block including a copy of the software instruction (see, for example, column 23, lines 16-19, which shows generating a recovery block that branches from the check instruction, and column 23, lines 42-51, which shows that the recovery block includes a copy of the load instruction).

With respect to claim 2 (original), the rejection of claim 1 is incorporated, and Ju further discloses that the software instruction comprises a load instruction (see, for example, column 22, lines 45-48, which shows the load instruction).

With respect to claim 3 (original), the rejection of claim 1 is incorporated, and Ju further discloses that the second location is positioned for instruction execution earlier than the first location is positioned for instruction execution (see, for example, column 22, lines 52-54, which shows that the second location is earlier than the first location).

With respect to claim 4 (original), the rejection of claim 1 is incorporated, and Ju further discloses that inserting a control speculative version of the software instruction at the second location comprises inserting a control speculative version of a load instruction (see, for example, column 23, lines 9-14, which shows inserting an eager-mode or control speculative version of the load instruction).

With respect to claim 10 (currently amended), Ju discloses an apparatus for preserving precise exceptions in code reordering (see, for example, column 22, lines 3-11, and column 23, line 67 to column 24, line 4), the apparatus comprising:

a processor to execute a plurality of software instructions (see, for example, column 15, line 54-67, which shows that the apparatus comprises such a processor);

a memory device operatively coupled to the processor to store the plurality of software instructions (see, for example, column 16, line 62 to column 17, line 3, which shows that the apparatus comprises such a memory device);

a control speculation module operatively coupled to the processor, the control speculation module being structured to insert a control speculative version of a software instruction into the plurality of software instructions in response to a determination that the software instruction is an excepting instruction (see, for example, column 22, lines 52-54, and column 23, lines 9-14, which shows a control speculation module that inserts an eager-mode or control speculative version of a load instruction into the plurality of instructions, and column 23, lines 27-36, which shows determining that the load instruction is an excepting instruction); and

an exception handler operatively coupled to the processor, the exception handler being structured to handle an exception associated with the control speculative version of the software instruction (see, for example, column 16, lines 29-46, which shows that the apparatus comprises such an exception handler).

With respect to claim 11 (original), the rejection of claim 10 is incorporated, and Ju further discloses that the control speculation module inserts a check instruction into the plurality of software instructions (see, for example, column 23, lines 14-16, which shows that the control speculation module inserts a check instruction).

With respect to claim 12 (original), the rejection of claim 11 is incorporated, and Ju further discloses that the control speculation module generates a recovery block which branches from the check instruction (see, for example, column 23, lines 16-19, which shows that the control speculation module generates a recovery block that branches from the check instruction).

With respect to claims 13 (currently amended) and 14-16 (original), the claims are directed to an apparatus that corresponds to the method of claims 1-4 (see the rejection of claims 1-4 above).

With respect to claims 17 (currently amended), 18 and 19 (original), the claims are directed to a machine readable medium that corresponds to the method of claims 1-4 (see the rejection of claims 1-4 above).

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 5-7 and 20-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ju in view of U.S. Patent No. 6,463,579 to McKinsey (art of record, "McKinsey") in view of U.S. Patent No. 7,065,750 to Babaian et al. (art of record, "Babaian").

With respect to claim 5 (currently amended), Ju discloses a method of preserving exceptions in code reordering (see, for example, column 22, lines 3-11, and column 23, line 67 to column 24, line 4), the method comprising:

receiving a plurality of instructions including a first instruction (see, for example, column 27, lines 58-65, which shows receiving a plurality of instructions including a first instruction).

Ju discloses moving a first instruction that is not an excepting instruction upward across a check instruction (see, for example, column 28, lines 10-51, which shows moving the first add instruction upward across a check instruction, and which shows that the load instruction is the excepting instruction), as in the following:

determining if the first instruction is an excepting instruction;

determining if the first instruction is to be moved upward across a check instruction;

Ju further discloses a second instruction that computes a previous value of a target register associated with an instruction (see, for example, column 27, lines 58-65, and column 28, lines 35-39, which shows that the second add instruction computes a previous value of the target register associated with the load instruction), but does not expressly disclose:

determining a second instruction in the plurality of instructions that computes a previous value of a target register associated with the first instruction when the first instruction is not an excepting instruction and the first instruction is to be moved upward across a check instruction;

determining if a source operand associated with the second instruction is available at the check instruction.

However, in an analogous art, McKinsey discloses generating recovery code for control and data speculation in every phase of compilation (see, for example, column 10, lines 45-52), and further discloses fully representing the definition and use instructions included in the recovery code (see, for example, column 7, lines 16-32). McKinsey further discloses determining if an instruction is data ready (see, for example, column 7, lines 45-50), so as to prevent any non-speculative instructions from executing with speculative operands (see, for example, column 7, lines 62-65).

Therefore, in view of McKinsey's teachings, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Ju such that the above determinations are made, so as prevent any non-speculative instructions from executing with speculative operands.

Ju in view of McKinsey further discloses inserting an instruction into a recovery block to restore the value of a target register (see, for example, Ju, column 28, lines 33-35), but does not expressly disclose:

inserting a third instruction into the plurality of instructions to save the value of the target register if the source operand associated with the second instruction is not available at the check instruction; and

inserting a fourth instruction into a recovery block to restore the value of the target register.

However, in an analogous art, Babaian discloses saving the value of a target register before an excepting instruction and restoring the value of the target register in the recovery code, so as to minimize the impact of an exception on execution performance (see, for example, column 7, lines 56-67).

Therefore, in view of Babaian's teachings, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Ju such that a third instruction is inserted into the plurality of instructions to save the value of the target register if the source operand associated with the second instruction is not available at the check instruction, and a fourth instruction is inserted into a recovery block to restore the value of the target register, so as to minimize the impact of an exception on execution performance.

With respect to claim 6 (original), the rejection of claim 5 is incorporated, and Ju in view of McKinsey in view of Babaian further discloses inserting a copy of the second instruction into the recovery block if the source operand associated with the second instruction is available at the check instruction (see, for example, Ju, column 28, lines 10-13 and 48-51, which shows inserting a copy of the second add instruction into the recovery block).

With respect to claim 7 (original), the rejection of claim 6 is incorporated, and Ju in view of McKinsey in view of Babaian further discloses that inserting a copy of the second instruction into the recovery block comprises inserting a copy of the second instruction into the recovery block ahead of a copy of the excepting instruction (see, for example, Ju, column 28, lines 33-35, which shows inserting a copy of the excepting load instruction into the recovery block, and note that in view of McKinsey, the copy of the second instruction is necessarily inserted ahead of the copy of the excepting load instruction to achieve the appropriate result).

With respect to claims 20 (currently amended), 21 and 22 (original), the claims are directed to a machine readable medium that corresponds to the method of claims 5-7 (see the rejection of claims 1-4 above).

7. Claims 8, 9, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ju in view of McKinsey.

With respect to claim 8 (currently amended), Ju discloses a method of preserving exceptions in code reordering (see, for example, column 22, lines 3-11, and column 23, line 67 to column 24, line 4), the method comprising:

receiving a plurality of instructions including a first instruction (see, for example, column 27, lines 58-65, which shows receiving a plurality of instructions including a first instruction).

Ju discloses moving a first instruction that is not an excepting instruction downward across a check instruction (see, for example, column 28, lines 10-51, which shows moving the second add instruction downward across a check instruction, and which shows that the load instruction is the excepting instruction), as in the following:

determining that the first instruction is an excepting instruction;

determining if the first instruction is to be moved upward across a check instruction; and

determining if the first instruction is to be moved downward across the check instruction.

Ju further discloses inserting a copy of the load instruction into a recovery block (see, for example, column 28, lines 33-35), but does not expressly disclose:

inserting a copy of the first instruction into a recovery block in response to determining that (i) the first instruction is not an excepting instruction, (ii) the first instruction is not to be moved upward across a check instruction, and (iii) the first instruction is to be moved downward across a check instruction.

However, in an analogous art, McKinsey discloses generating recovery code for control and data speculation in every phase of compilation (see, for example, column 10, lines 45-52), and further discloses fully representing the definition and use instructions included in the recovery code (see, for example, column 7, lines 16-32). McKinsey further discloses inserting the appropriate instructions into the recovery code for an instruction that crosses a check instruction (see, for example, column 8, lines 4-20).

Therefore, in view of McKinsey's teachings, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Ju such that a copy of the first instruction is inserted into a recovery block in response to determining that (i) the first instruction is not an excepting instruction, (ii) the first instruction is not to be moved upward across a check instruction, and (iii) the first instruction is to be moved downward across a check instruction, so as to generate the appropriate recovery code for the plurality of instructions.

With respect to claim 9 (original), the rejection of claim 8 is incorporated, and Ju in view of McKinsey further discloses that inserting a copy of the first instruction into the recovery block comprises inserting a copy of the first instruction into the recovery block ahead of a copy of the excepting instruction (see, for example, Ju, column 28, lines 33-35, which shows inserting a copy of the excepting load instruction into the recovery block, and note that in view of McKinsey, the copy of the first instruction is necessarily inserted ahead of the copy of the excepting load instruction to achieve the appropriate result).

With respect to claims 23 (currently amended) and 24 (original), the claims are directed to a machine readable medium that corresponds to the method of claims 8 and 9 (see the rejection of claims 8 and 9 above).

Conclusion

8. Applicant's amendment necessitated any new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER